Padrão CMSIS e Biblioteca TivaWare

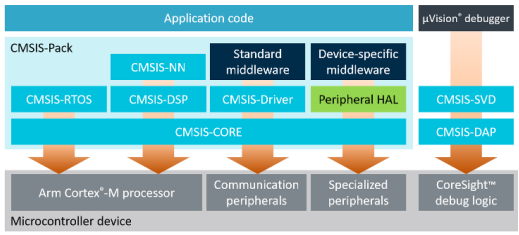
Prof. Hugo Vieira Neto
2020/1

Objetivo

- Estudar as principais características do padrão CMSIS e da biblioteca TivaWare:
  - CMSIS-Core
  - Estrutura de arquivos de um projeto
  - Nomes padrão para exceções do sistema
  - Camadas de abstração de hardware

CMSIS

Cortex Microcontroller Software Interface Standard



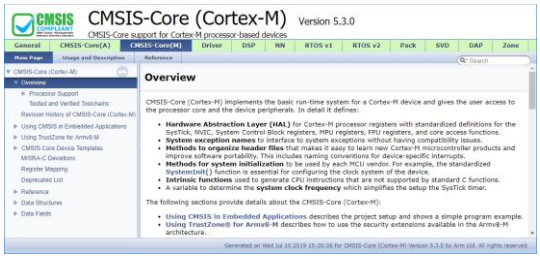
Principais Componentes

- CMSIS-Core
  - API para núcleos Cortex-M ou Cortex-A
- CMSIS-Driver
  - Drivers genéricos para sistemas de comunicação, sistemas de arquivos, interfaces gráficas, etc.
- CMSIS-DSP
  - Biblioteca para processamento de sinais digitais
- CMSIS-NN
  - Biblioteca para implementação de redes neurais
- CMSIS-RTOS
  - API para sistemas operacionais em tempo real

Componente CMSIS-Core

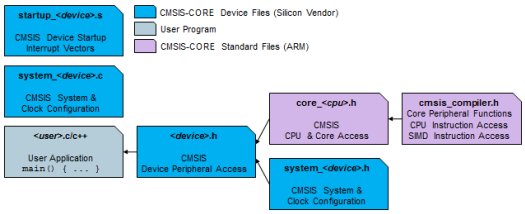
- Sistema básico, que provê acesso ao núcleo do processador Cortex-M e aos periféricos do dispositivo, definindo:
  - Camada de abstração de hardware
  - Nomes para as exceções do sistema
  - Funções intrínsecas [instruções do core]
  - Função **`SystemInit`** [inicialização do sistema]
  - Variável **`SystemCoreClock`** [frequência de clock]

Documentação



http://www.keil.com/pack/doc/cmsis/Core/html/index.html

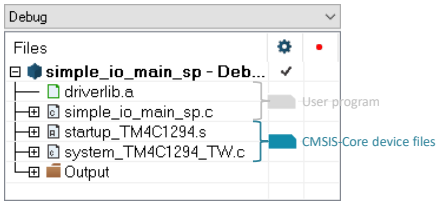
Estrutura de Arquivos



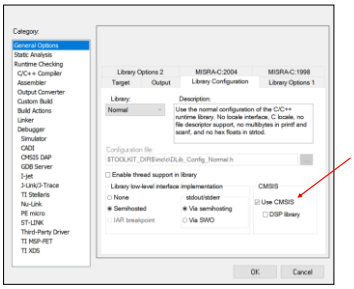
Componente CMSIS-Core

- Arquivos (Cortex-M):
  - **`startup_<device>.c`** ou **`startup_<device>.s`**
    - Tabela de vetores de exceção
  - **`system_<device>.c`** e **`system_<device>.h`**
    - Inicialização (clock, FPU, WDT, bus wait states, etc.)
  - **`<device>.h`**
    - Camada de abstração de hardware (HAL) dos registradores do núcleo e dos registradores dos periféricos do dispositivo (device)

Estrutura de Arquivos no IAR



Opções do Projeto no IAR



Nomes das Exceções do Sistema

- **`Reset_Handler`**
- `NMI_Handler`
- `HardFault_Handler`
- `MemManage_Handler`
- `BusFault_Handler`
- `UsageFault_Handler`
- **`SVC_Handler`**
- `DebugMon_Handler`
- **`PendSV_Handler`**
- **`SysTick_Handler`**

Camada de Abstração de Hardware

- Estruturas de dados contendo ponteiros para os registradores de cada bloco do dispositivo
- Arquivo header único (`TM4C1294NCPDT.h`)
- Exemplo de uso:
  - `GPIOK->DIR` acessa o registrador `DIR` do bloco `GPIOK` (endereço `0x40061400`)
  - Ver Seção 10.5 do datasheet do dispositivo TM4C1294NCPDT

Informações do Datasheet

GPIO Base Adresses

- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (AHB): 0x4005.B000
- GPIO Port E (AHB): 0x4005.C000
- GPIO Port F (AHB): 0x4005.D000
- GPIO Port G (AHB): 0x4005.E000
- GPIO Port H (AHB): 0x4005.F000
- GPIO Port J (AHB): 0x4006.0000
- GPIO Port K (AHB): 0x4006.1000
- GPIO Port L (AHB): 0x4006.2000
- GPIO Port M (AHB): 0x4006.3000
- GPIO Port N (AHB): 0x4006.4000
- GPIO Port P (AHB): 0x4006.5000
- GPIO Port Q (AHB): 0x4006.6000

GPIO Register Map

Offset	Name	Type
~~0x000~~	GPIODATA	RW
0x400	GPIODIR	RW
0x404	GPIOIS	RW
0x408	GPIOIBE	RW
0x40C	GPIOIEV	RW
0x410	GPIOIM	RW
0x414	GPIORIS	RO
0x418	GPIOMIS	RO
0x41C	GPIOICR	W1C

Biblioteca TivaWare

- Também provê uma camada de abstração
- Macros para acesso aos registradores
- Múltiplos arquivos header com os endereços de base de cada bloco e os deslocamentos de cada registrador (`inc/hw_*.h`)
- Importante: deve-se definir o dispositivo alvo da compilação (p. ex. `PART_TM4C1294NCPDT`)
  - Options → C/C ++ Compiler → Preprocessor

Biblioteca TivaWare

- Exemplo de acesso a registrador:
  - `HWREG(GPIO_PORTK_BASE + GPIO_O_DIR)`
- A macro `HWREG` é definida em `hw_types.h`
- `GPIO_PORTK_BASE` = `0x40061000` definido em `hw_memmap.h`
- `GPIO_O_DIR` = `0x00000400` definido em `hw_gpio.h`

TivaWare Peripheral Driver Library

- API de device drivers para os periféricos do dispositivo (`driverlib/*.h`)
- Definições de constantes para configuração dos periféricos
- Exemplos (`driverlib/gpio.h`):
  - `GPIO_DIR_MODE_IN  = 0x00000000`
  - `GPIO_DIR_MODE_OUT = 0x00000001`
  - `GPIO_DIR_MODE_HW  = 0x00000002`

Legibilidade de Código

- Qual das versões equivalentes de código apresentadas a seguir é mais legível?
  - `GPIOK->DIR |= GPIO_PIN_1`
  - `GPIOK->DIR |= 0x00000002`
- Ver "TivaWare Peripheral Driver Library User's Guide"
  - `GPIOPinTypeGPIOInput(GPIO_PORTK_BASE, GPIO_PIN_1)`
- Ver `inc/hw_memmap.h` e `driverlib/gpio.h`

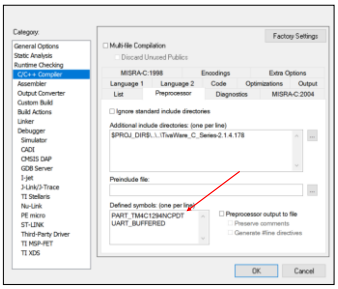
Exercício

- Selecione o projeto "simple_uart" a partir da área de trabalho "EK-TM4C1294XL_IAR".
- Use um aplicativo emulador de terminal, como o Tera Term, com uma conexão serial com "COMn: Stellaris Virtual Serial Port" para se comunicar com o kit EK-TM4C1294XL.

Exercício

- Note que o projeto faz uso do arquivo "uartstdio.c" em "TivaWare_C_Series-2.1.4.178\utils"
  - A macro `PART_TM4C1294NCPDT` deve ser definida para a configuração adequada dos terminais RX e TX
  - A macro `UART_BUFFERED` deve ser definida se buffers controlados por interrupções forem usados

Exercício



Exercício

- Uma rápida olhada nos arquivos "uartstdio.h" e "uartstdio.c" fornece informações importantes:
  - `UART_RX_BUFFER_SIZE` define o tamanho do buffer de recepção, que pode ser substituído
  - `UART_TX_BUFFER_SIZE` define o tamanho do buffer de transmissão , que pode ser substituído
  - `UARTStdioIntHandler` é o nome da rotina de tratamento de interrupção da UART, que é **diferente** do padrão em "startup_TM4C1294.s"

Exercício

- Inspecione o arquivo "simple_uart.c":
  - O que a função `UARTInit` faz?
  - O que acontece se alguém remover a definição de `PART_TM4C1294NCPDT` das opções do Preprocessador do Compilador C/C++ no IAR?
  - Como foi tratado problema do nome da rotina de tratamento de interrupção da UART ter nome diferente do padrão em "startup_TM4C1294.s"? Existem soluções alternativas? Quais são os possíveis problemas com elas?

Exercício

- Faça alterações no código para que a rotina de tratamento de interrupção do SysTick, além de alternar o estado do LED D1, também envie "SysTick_Handler\n" via UART
- Observe o que acontece no Tera Term – as mensagens recebidas são sincronizadas com as alterações no estado do LED D1?
- Obs: a taxa de transmissão deve ser definida em 9600 bps

Exercício

- Pause o programa no IAR e inspecione essas variáveis:
  - `g_ui32UARTTxWriteIndex`
  - `g_ui32UARTTxReadIndex`
  - `g_pcUARTTxBuffer`
- O que essas variáveis lhe dizem? Veja os comentários para "output ring buffer" em "uartstdio.c"
- O que acontece se a taxa de transmissão for reduzida para 600 bps? E para 300 bps? Não se esqueça de redefinir os dois lados da conexão (kit **e** Tera Term)